# Mobile Heart Monitor

Autor: Necula Alexandru

Grupa: 336CA

## Introducere

Proiectul consta in realizarea unui aparat care masoara frecventa cardiaca, cat si nivelul de oxigenare din sange.

Acesta ar putea fi util pentru persoanele cu probleme cardiace sau în cazul unei infectii respiratorii precum COVID-19, în care saturația oxigenului din sânge scade la un nivel periculos.

## Descriere generală

Interfata cu utilizatorul consta in:

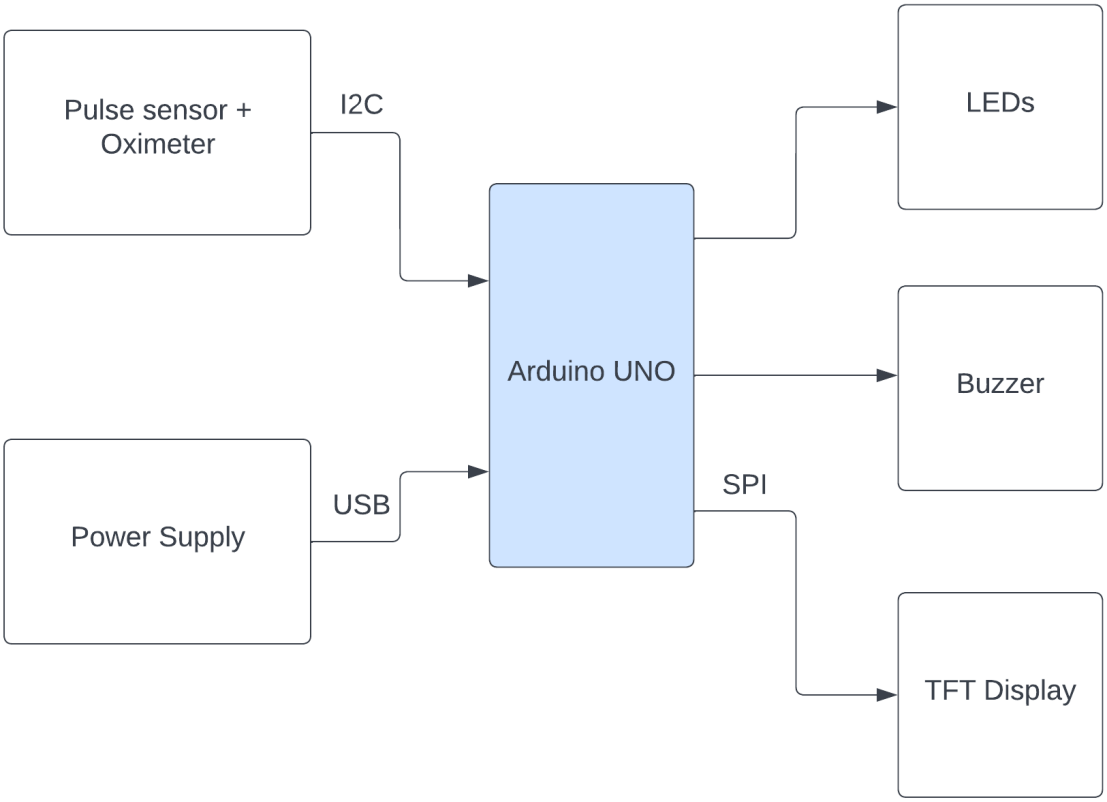
- display TFT pe care se afiseaza informatiile
- LED si buzzer care se activeaza cu fiecare bataie a inimii
- LED RGB care semnalizeaza sanatatea inimii (pe baza datelor de la senzor)

Pentru a afisa informatiile, degetul trebuie pus pe senzor, iar dupa cateva secunde, vor fi afisate informatii, semnalate prin LED-ul si buzzer-ul care se vor activa cu fiecare bataie a inimii.

Culoarea LED-ului RGB va fi schimbata in functie de nivelul pulsului

- VERDE: puls < 100
- GABLEN: puls < 120
- ROSU: puls > 120
- ALB: puls in curs de calcul

### Schema bloc




# Hardware Design

## Lista componente

| Nume | Număr Piese |
|---|---|
| Arduino UNO R3 | 1 |
| Breadboard | 1 |
| Ecran TFT ST7789 1.3” 240×240 | 1 |
| Senzor puls MAX30100 | 1 |
| LED rosu | 1 |
| LED RGB | 1 |
| Buzzer | 1 |
| Rezistor 1kΩ | 5 |
| Rezistor 460Ω | 4 |
| Rezistor 330Ω | 4 |
| Jumper | 18 |

## Schema electrica

Placuta comandata care include senzorul are o problema de design, respectiv liniile de SDA si SCL sunt HIGH la tensiunea de 1.8V, si nu 3.3V, astfel incat Arduino UNO nu poate comunica cu senzorul [1)]. Am modificat placuta printr-un jumper astfel incat SDA si SCL sunt alimentate de la 3.3V.

Display-ul TFT nu este compatibil cu tensiunea de 5V a pinilor de la Arduino UNO. Pe pinii folositi de SPI am instalat un divizor de tensiune folosind rezistori de 460Ω si 1kΩ, astfel incat tensiunea finala este de aproximativ 3.3V [2)]

# Software Design

## Mediul de dezvoltare

- **Visual Studio Code + extensia PlatformIO** → dezvoltarea codului și încărcarea acestuia pe Arduino
- **Autodesk Eagle** → realizarea schemei electrice
- **LucidChart** → realizarea schemei bloc

## Biblioteci folosite

- **SPI** [3)]
- **Wire** (I2C communication) [4)]
- **MAX30100** (pulse sensor) [5)]
- **Adafruit GFX** (core graphics) [6)]
- **Adafruit BusIO** (I2C communication) [7)]
- **Adafruit ST7789** (specific TFT driver) [8)]

## Structura codului

Senzorul de puls are 3 stari:

- **WAITING** → degetul nu se afla pe senzor; se afiseaza mesajul “Rest your finger on the sensor”
- **INITIALIZING** → frecventa cardiaca este in curs de calcul; se afiseaza mesajul “Please wait…”
- **WORKING** → display-ul afiseaza datele colectate; se afiseaza pulsul si oxigenarea sangelui pe display

**Initializari** - inainte de orice functie, initializam urmatoarele variabile:

- *pox* → clasa care defineste senzorul de puls-oximetru
- *tft* → clasa care defineste display-ul tft
- *lastState* → initializat cu WAITING, folosita pentru a tine cont cand actualizam display-ul
- *prevHeartRate* → initializat cu 0, folosita pentru a determina daca calculul pulsului este stabil
- *tsLastReport* → initializat cu 0, folosita pentru a actualiza display-ul la intervale determinate

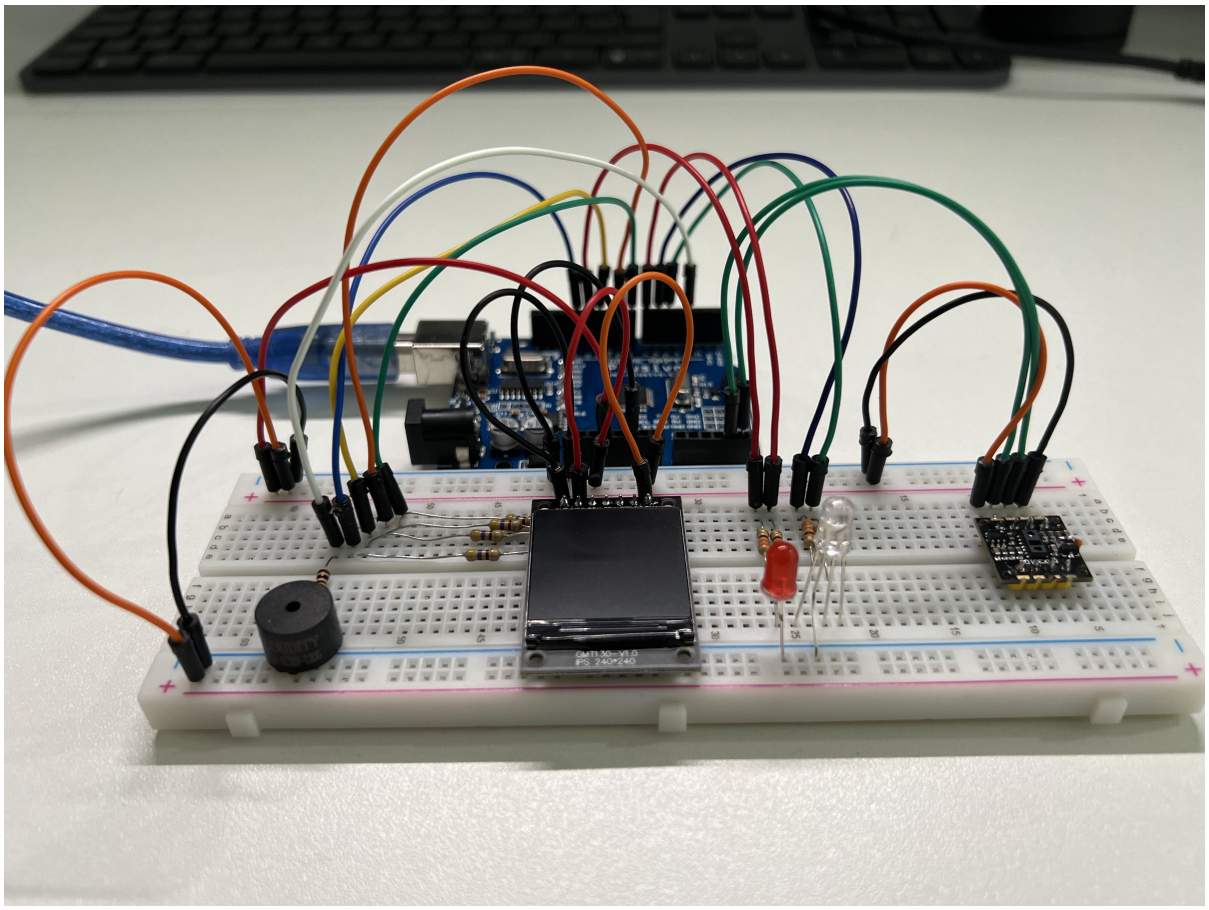
Senzorul puls-oximetru trebuie interogat cat mai rapid, altfel datele din buffer-ul sau se pierd. Conform bibliotecii utilizate, actualizarea ar trebui facuta la aproximativ 100Hz. De aceea, in majoritatea functiilor implementate am apelat *pox.update()*

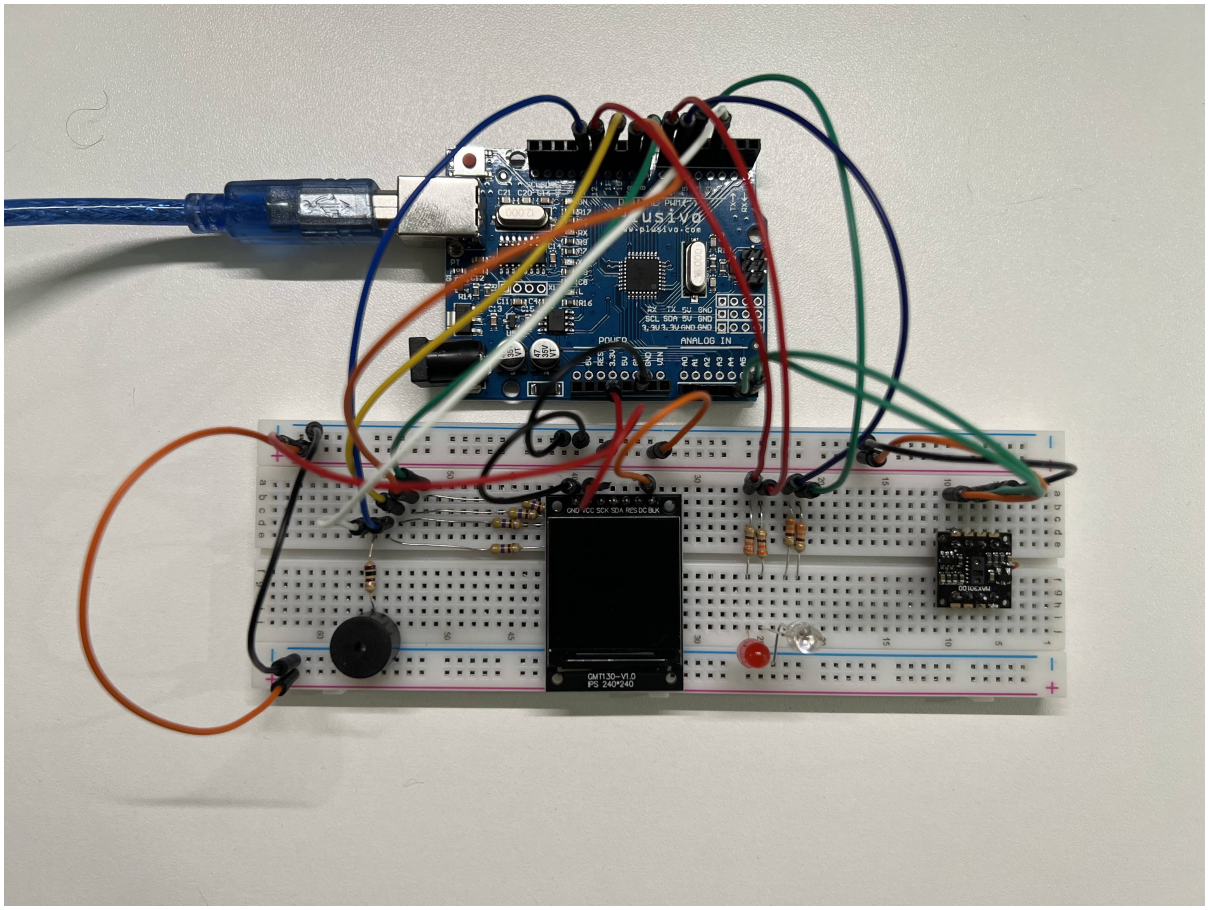
### Functii

- *setup()* → initializarea pinilor pentru buzzer si LED-uri, initializarea display-ului TFT, initializarea senzorului de puls, inregistrarea functie de callback pentru fiecare puls
- *loop()* → logica principala a programului: apeleaza *pox.update()*; la fiecare 2 secunde calculeaza pulsul si SpO2 si afiseaza informatii relevante pe display in functie de aceste valori
- *RGB_color(red, green, blue)* → seteaza pinii LED-ului RGB cu valorile date ca parametri
- *setRGBColor(spO2, heartRate)* → apeleaza *RGB_color* cu valori in functie de puls si SpO2
- *onBeatDetected()* → functie callBack; porneste buzzer-ul si LED-ul rosu de fiecare data cand detectam un puls
- *printOnDisplay(x, y, whatToPrint)* → afiseaza pe display mesajul/valoarea dorita; deoarece aceasta operatie dureaza mai mult timp, apeleaza *pox.update()*
- *clearDisplay()* → diferit de implementarea din biblioteca Adafruit, pentru a permite interogarea senzorului de puls; printeaza cate un spatiu pe ecran pentru a sterge continutul anterior

## Rezultate Obţinute

### Circuit final





### Demo

YouTube link

## Concluzii

Acest proiect m-a facut sa invat mai multe despre cum pot crea un dispozitiv cu utilitate practica folosind un microcontroller si cum pot programa un ansamblu de piese pentru a ajunge la rezultatul dorit.

De asemenea, am invatat ca este foarte important sa citesti cu atentie datasheet-ul componentelor pe care doresti sa le comanzi pentru a te asigura ca sunt compatibile cu microcontroller-ul folosit. In cazul meu, a fost nevoie de pasi suplimentari pentru a asigura compatibilitatea, respectiv divizorul de tensiune pentru LCD si jumper-ul lipit pe senzorul de puls.

## Download

mobile_heart_rate_monitor.zip

# Jurnal

- **14.04.2022**: alegere tema proiect
- **15.04.2022**: comanda piese
- **21.04.2022**: publicare descriere wiki
- **29.04.2022**: finalizare software si asamblare hardware
- **30.04.2022**: finalizare wiki, publicare documentatie completa

# Bibliografie/Resurse

Export to PDF

[1] https://how2electronics.com/interfacing-max30100-pulse-oximeter-sensor-arduino/
[2] https://simple-circuit.com/arduino-st7789-ips-tft-display-example/
[3] https://www.arduino.cc/en/reference/SPI
[4] https://www.arduino.cc/en/reference/Wire
[5] https://github.com/oxullo/Arduino-MAX30100
[6] https://github.com/adafruit/Adafruit-GFX-Library
[7] https://github.com/adafruit/Adafruit_BusIO
[8] https://github.com/cbm80amiga/Arduino_ST7789_Fast